**EE551000 System Theory**

**HW5: Solving cartpole using RL**

羅文生

107064426

**Goal and Todo:**

The goal of this test is to solve a classic control problem by implementing multiple reinforcement learning methods learned in this class. In this test, we have to implement the algorithms all by your own.

• **Environment**

o Cartpole-v1 (OpenAI Gym library) .

• **The observation space (state space) is:**

o Cart position (max: 4.8, min: -4.8)

o Cart velocity (max: Inf, min: -Inf)

o Pole Angle (max: 24 deg, min: -24 deg)

o Pole velocity at tip (max: Inf, min: -Inf)

• **The action space is:**

o Push cart to left (0)

o Push cart to right (1)

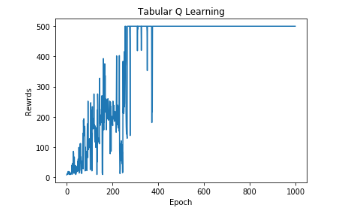
• Taking each step would earn reward +1.

**Implementation**

**Google colab was used for this assignment**

• Tabular Q-learning method. (how to transform continuous state to discrete state?)

To handle continuous state, we must transform the observed continuous information such as pole velocity, pole angle, cart velocity, cart position to discrete values. For that I chose to use state aggregation to discretize the state. The idea is that we want information that is closer together to be in just one state. After trial and error, the cart position can be represented with 4 values, the cart velocity can be represented with two values, the pole angle can be represented with 6 values and the pole velocity is represented with 4 values. After converting state to discrete values, we can use the original Q-learning algorithm to complete the task. But, two new values were added. We have a decay value because we want the agent to explore less after finding the optimal policy. The decay variable reduces exploration t each episode. We have a penalty. I have found it best that the agent tends to learn faster when it is punished for make the wrong choice. The penalty is -1000.

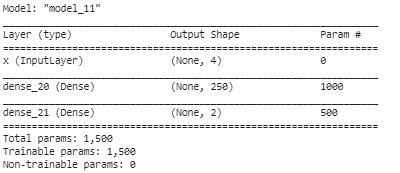


We can see that the agent is able to reach the optimal policy at around 250 steps. We can see the effect of the decay variable on epsilon as when the number of episodes gets to 500, there is less exploration and the agent tends to act greedily there no mistakes were made. It took a little less than a minute to train.

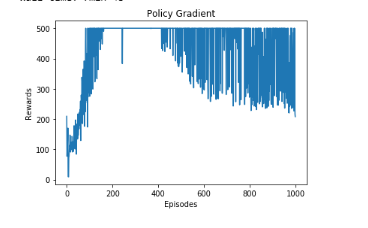
• Policy Gradient method.

Due to the continuous nature of this problem, I have use keras to train the agent. The algorithm was taken from the book. The number of hidden states was 250 and the learning rate of 0.1.

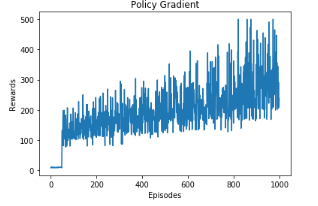
This is the resulting model.



It has one input layer which are the four observation values the we get after each action, and also 2 dense layers, with the second having 250 components to be able to extract more information. With a batch of 1, we get this result.



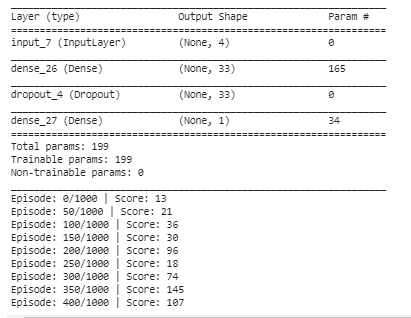
The agent was able to perform much faster than the q-learning agent and it reached its optimal policy at about 100 steps. It took about 3 minutes to train.



This is the result with a batch size of 50.

• Actor-Critic method.

The Actor critic method also required the use of neural networks to simplify this task. The important thing to remember is that we want the critic to learn much faster than the actor as a result, we have that the critic has a bigger learning rate than the actor. The algorithm flows that of the book.



We see that the other algorithms seem to perform faster than this method, but it could have a lot to do with the fact that there are two weights to train. This model was taking too long to train so I was unable to provide the plot.

With this assignment, we deal with a much different task than what we are used to. Usually we have a prior knowledge of the states that we are going to encounter but now we have no prior knowledge of the states before training. As a result, with a small number of episodes we risk not being able to get to enough state to be able to find the optimal policy. The is a big problem for ANN’s as the require a lot of training to fine-tune their weights to get the best number of rewards. It has hard to say which algorithm performs the best as it depends on the needs at hands. Q-learning is faster, but it requires a lot of preprocessing before we can get the best out of it while the other two methods require a lot more time for training and are more computationally expensive. Also, the results for training these methods with ANN are not always going to be consistent as there are many ways to train these algorithms has having hidden dimensions, having more layers or using a different batch size.